Data Structures

1. class Queue {
2. constructor(limit) {
3. this.items = [];
4. this.maxSize = limit;
5. }
6. enqueue(value) {
7. if(this.items.length == this.maxSize) {
8. console.log("Stack is Full");
9. return;
10. }
11. this.items.push(value);
12. console.log("Enqueued -> " + value);
13. }
14. isEmpty() {
15. return this.items.length == 0;
16. }
17. dequeue() {
18. if(this.isEmpty()) {
19. console.log("Stack is Empty");
20. return -1;
21. }
22. return this.items.shift();
23. }
24. front() {
25. if(this.isEmpty()) {
26. console.log("Stack is Empty");
27. return -1;
28. }
29. return this.items[0];
30. }
31. rear() {
32. if(this.isEmpty()) {
33. console.log("Stack is Empty");
34. return -1;
35. }
36. return this.items[this.items.length-1];
37. //[1,2,3,4,5,6,7] -> 7
38. // 0 1 2 3 4 5 6
39. }
40. size() {
41. return this.items.length;
42. }
43. print() {
44. while(!this.isEmpty()) {
45. console.log(this.dequeue());
46. }
47. }
48. }
49. var q = new Queue(10);
50. q.enqueue(1);
51. q.enqueue(2);
52. q.enqueue(3);
53. q.enqueue(4);
54. q.enqueue(5);
55. console.log(q.dequeue());
56. q.enqueue(6);
57. q.enqueue(7);
58. q.enqueue(8);
59. console.log(q.dequeue());
60. q.enqueue(9);
61. q.enqueue(10);
62. q.enqueue(11);
63. q.enqueue(12);
64. q.print();